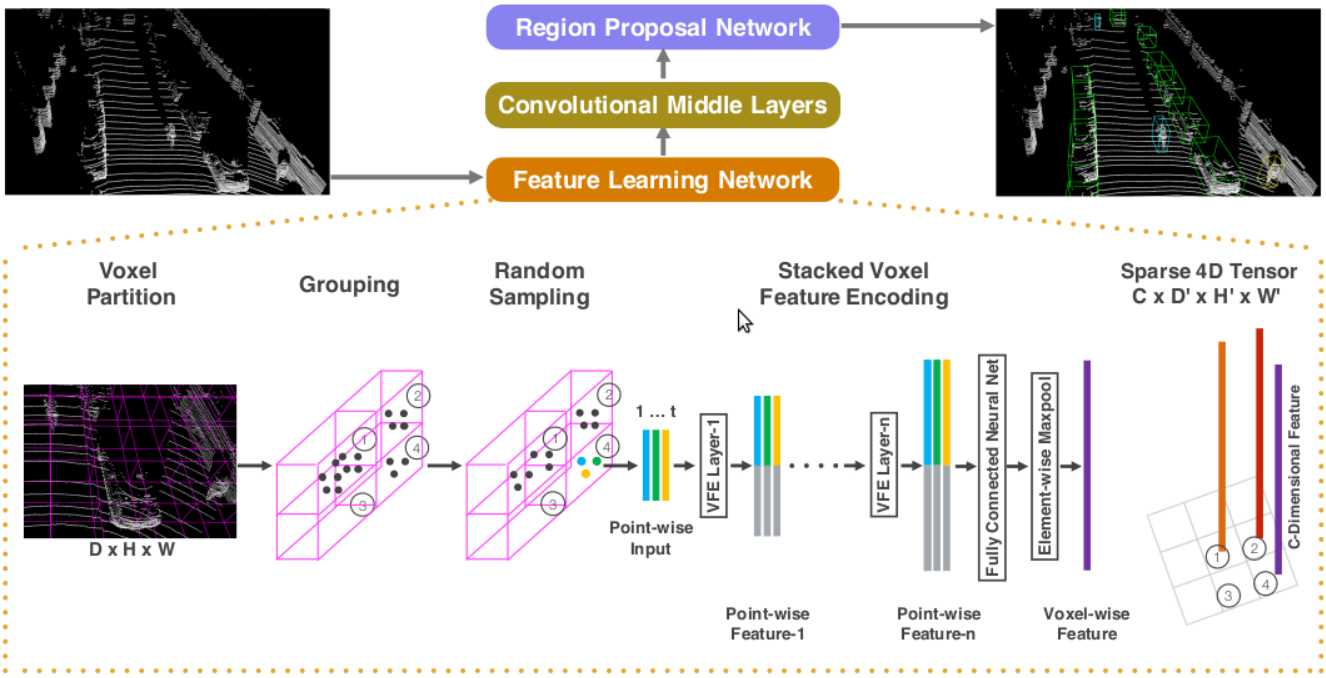
VoxelNet

To interface a highly sparse稀疏 LIDAR point cloud with a region proposal network(RPN区域建议网络),most existing efforts have focused on hand-crafted手工制作 feature representations特征表示，for example, a bird's eye view projectiong鸟瞰图.

In this work,we remove the need of manual feature engineering for 3D point clouds and purpose VoxelNet, a generic 3D detection network that unifies feature extraction and bounding box prediction into a single stage, end-to-end trainable deep network.

Specially,VoxelNet divides a point cloud into equally spaced 3D voxels and transforms a group of points within each voxel into a unified feature representation through the newly introduced voxel feature encoding(VFE) layer.In this way,the point cloud is encoded as a descriptive volumetric representation, which is then connected to a RPN to generate detections.

Our network learns an effective discriminative representation区分性表示 of objects with various geometrics, leading to encouraging results in 3D detection of pedestrains and cyclists, based on only Lidar.



VoxelNet architecture. The feature learning network takes a raw point cloud as input, partitions the space into voxels, and transforms points within each voxel to a vector representation characterizing the shape information.The space is represented as a sparse 4D tensor.The convolutional middle layers processes the 4D tensor to aggregate spatial context 聚合空间语境.Finally,a RPN generates the 3D detection.

Scalling up 3D feature learning networks to orders of magnitude more points and to 3D detection tasks are the main challenges that we address in this paper.

We present VoxelNet, a generic 3D detection framework that simultaneously同时 learns a discriminative判别性 feature representation from point clouds and predicts accurate 3D bounding boxes, in an end-to-end fashion方式.

We design a novel新颖的 voxel feature encoding(VFE) layer,which enables inter-point interaction点间交互 within a voxel,by combining point-wise features逐点特征 with a locally aggregated feature聚合特征.

Stacking 堆叠 multiple VFE layers allows learning complex features for characterizing表征 local 3D shape information.

Specially, VoxelNet divides the point cloud into equally spaced 3D voxel等距的３Ｄ体素 ,encodes each voxel via stacked VFE layers, and then 3D convolution furture进一步 aggregate聚合 local voxel features, transforming the pointcloud into a high-dimensional volumetric体积 representation.

Finally,a RPN consumes the volumetric represetation and yields产生 the detection result.

This efficient algorithm benefits both from the sparse稀疏的 point structure and efficient parallel processing on the voxel grid体素网格的高效并行处理.

VoxelNet Architecture

The proposed VoxelNet consists of three functional blocks:(1) Feature learning network,(2) Convolutional middle layers, and (3) Region proposal network

Feature Learning Network

Voxel Partition Given a point cloud, we subdivide the 3D space into equally spaced voxels as shown in Figure 2.Suppose the point cloud encompasses包含 3D space with range D,H,W（点云尺寸） along the Z,Y,X axes respectively.We define each voxel of size vD,vH,and vW（体素尺寸） accordingly.The resulting所生成的 3D voxel grid is of size D'=D/vD, H'=H/vH, W'=W/vW（点云包含体素个数）.Here, for simplicity,we assume D,H,W are a multiple of vD,vH,vW.我们假设D,H,W是vD, vH, vW的倍数。

Grouping We group the points according to the voxel they reside in我们根据它们所在的体素对点进行分组.Due to factors such as distance,occlusion遮挡,object's relative pose物体的相对姿势,and non-uniform不均匀 sampling,the LiDAR point cloud is sparse and highly variable高度可变的 point density点密度 throughout the space.Therefore, after gouping, a voxel will contain a variable number of points.

Random Sampling Typically a high-definition LiDAR point cloud is composed of ~100k points. Directly processing all the points not only imposes加强 increased memory/efficiency burdens负担 on the computing platform, but also highly variable point density throughout the space might bias the detection.To this end因此,we randomly sample a fixed number, T,of points from those voxels containing more than T points.This sampling strategy战略 has two purpose,(1)computational saving;and (2)decrease the imbalance of points between the voxels which reduces the sampling bias, and adds more variation变化 to training.

Stacked Voxel Feature Encoding

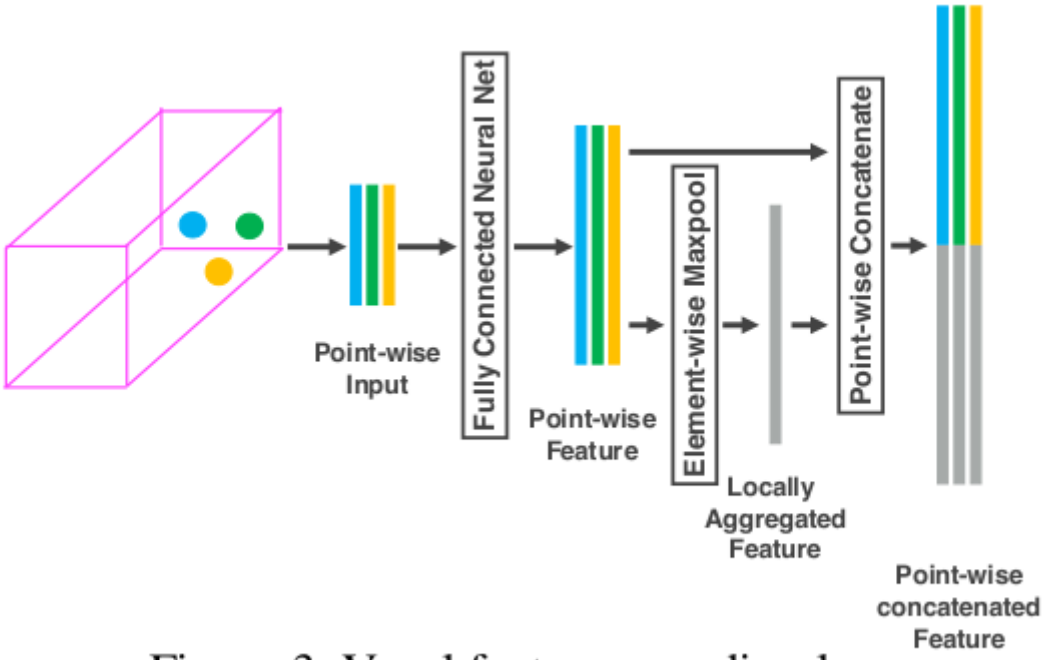


Figure 3. Voxel feature encoding layer.

The key innovation革新 is the chain链 of VFE layers.For simplicity, Figure 2 illustrates the hierachical阶级式 feature encoding process for one voxel.Without loss of generality概要,(在不失一般性的前提下)we use VFE Layer-1 to describe the details in the following paragraph段落.Figure 3 shows the architecture for VFE Layer-1.

Denote表示 $\mathbf{V} = \{\mathbf{p}_i = [x_i, y_i, z_i, r_i]^T \in \mathbb{R}^4\}_{i=1\ldots t}$ as a non-empty voxel containing t≤ T LiDAR points,where $\mathbf{p}_i$ contains XYZ coordinates for the i-th point and r_i is the received reflectance反射率.We first compute the local mean局部均值 as the centroid质心 of all the points in V, denoted as (v_x, v_y, v_z).Then we augment增加 each point $\mathbf{p}_i$ with the relative offset w.r.t. the centriod and obtain the input feature set $\mathbf{V}_{\text{in}} = \{\hat{\mathbf{p}}_i = [x_i, y_i, z_i, r_i, x_i - v_x, y_i - v_y, z_i - v_z]^T \in \mathbb{R}^7\}_{i=1\ldots t}$, Next, each $\hat{\mathbf{p}}_i$ is transformed through the fully connected network(FCN) into a feature space,where we can aggregate汇总 information from the point features $\mathbf{f}_i \in \mathbb{R}^m$ to ecode the shape of the surface contained within the voxel.The FCN is composed of a linear layer, a batch normalization(BN批量标准化) layer and a rectified纠正的 linear unit(ReLU) layer.After obtaining point-wise feature representations, we use element-wise MaxPooling across all $\mathbf{f}_i$ associated to V to get the locally aggregated feature $\tilde{\mathbf{f}} \in \mathbb{R}^m$ for $\mathbf{V}$. Finally, we augment each $\mathbf{f}_i$ with $\tilde{\mathbf{f}}$ to form the point-wise concatenated 级联的 feature as $\mathbf{f}_i^{out} = [\mathbf{f}_i^T, \tilde{\mathbf{f}}^T]^T \in \mathbb{R}^{2m}$ Thus we obtain the output feature set $\mathbf{V}_{\text{out}} = \{\mathbf{f}_i^{out}\}_{i\ldots t}$.All non-empty voxels are ecoded in the same way and they share the same set of parameters in FCN.

We use $\text{VFE-i}(c_{in}, c_{out})$ to represent the i-th VFE layer that transforms input features of dimension c_{in} into output features of dimension c_{out}. The linear layer learns a matrix of size $c_{in} \times (c_{out}/2)$, and the point-wise concatenation yields the output of dimension c_{out}.

Because the output feature combines both point-wise features and locally aggregated feature, stacking VFE layers encodes point interactions互动 within a voxel and enables the final feature representation to learn descriptive shape information. The voxel-wise feature is obtained by transforming the output of VFE-n into $\mathbb{R}^C$ via FCN and applying element-wise Maxpool where C is the dimension of the voxel-wise feature, as shown in Figure2.

Saparse Tensor Representation稀疏张量表示 By processing only the non-empty voxels, we obtain a list of voxel features,each uniquely独特的 associated to the spatial空间 coordinates of a pictular non-empty voxel.The obtained list of voxel-wise features can be represented as a sparse 4D tensor, of size C x D' x H' x W' as shown in Figure2.Although the point cloud contains ~100k points, more than 90% of voxels typically are empty.Representing non-empty voxel features as a sparse tensor greatly reduce the memory usage and computation cost during backpropagation反向传播, and it is a critical step in our efficient implementation.

Convolutional Middle Layers

We use ConvMD$(c_{in}, c_{out}, \mathbf{k}, \mathbf{s}, \mathbf{p})$ to represent an M-dimensional convolution operator where c_{in} and c_{out} are the number of input and output channels, k,s, and p are the M-dimensional vectors correspoinding to kernel size, stride size and padding填充 size respectively.When the size across the M-dimensions are the same, we use a scalar to represent the size e.g. k for k=(k,k,k).

Each convolutional middle layer applies 3D convolution,BN layer, and ReLU layer sequentially依次.The convolutional middle layers aggregate voxel-wise features within a progressively逐步 expanding扩大的 receptive接收 field，adding more context to the shape description.The detialed sizes of the filters in the convolutional middle layers are explained in Section 3.

Region Proposal Network

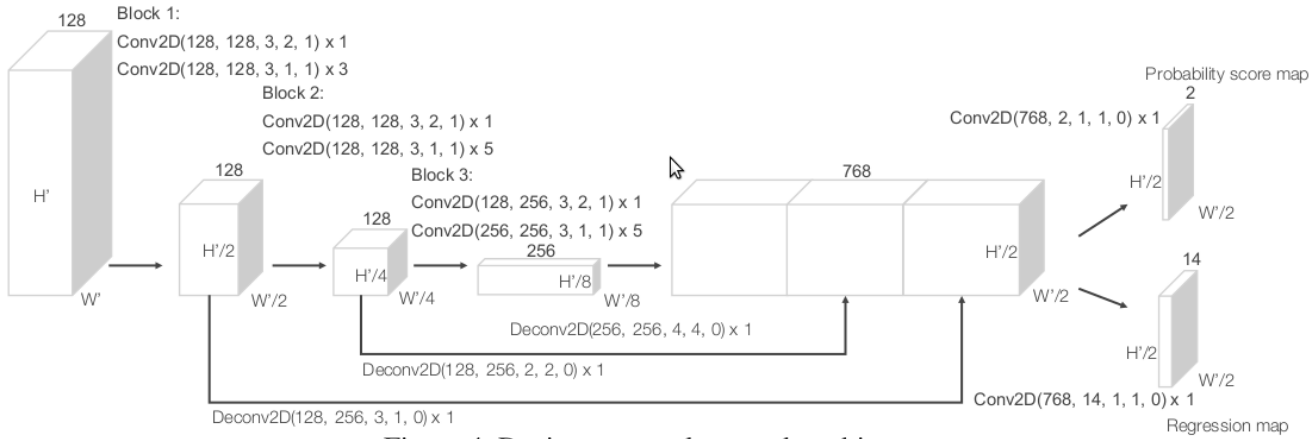


Figure 4. Region proposal network architecture.

Recently, region proposal提案 networks have become an important building block of top-performing object detection frameworks. In this work,we make servel key modifications to the RPN architecture proposed in

[34], and combine it with the feature learning network and convolutional middle layers to form an end-to-end trainable pipeline管道.

The input to our RPN is the feature map provided by the convolutional middle layers.The architecture of this network is illustrate in Figure 4.The network has three blocks of fully convolutional layers.The first layer of each block downsamples the feature map by half via a convolution with a stride size of 2,followed by a sequence of convolutions of stride 1(xq means q applications of the filter).After each convolution layer,BN and ReLU operations are applied.We then upsample the output of every block to a fixed size and concatanate to construct the high resolution feature map. Finally, this feature map is mapped to the desired learning targets:(1) a probability score map and (2) a regression map. R P N的输入是卷积中间层提供的特征图。该网络的体系结构如图 4 所示。该网络具有三个完全卷基层的块。每个块的第一层通过步幅为 2 的卷积对特征图进行一半下采样，然后是步幅 1 的卷积序列（xq 表示滤波器的 q 个应用）。在每个卷基层之后，BN 和ReLU 操作被应用。然后，我们将每个块的输出上采样到固定大小，并汇总以构建高分辨率特征图。最后，将此特征图映射到所需的学习目标：（1）概率分数图和（2）回归图。

Loss Function

Let $\{a_i^{\text{pos}}\}_{i=1\dots N_{\text{pos}}}$ be the set of N_{pos} positive anchors锚点 and $\{a_j^{\text{neg}}\}_{j=1\dots N_{\text{neg}}}$ be the set of N_{neg} negative anchors. We parameterize a 3D ground truth box as $(x_c^a, y_c^a, z_c^a, l^a, w^a, h^a, \theta^a)$,where x_c^g, y_c^g, z_c^g represent the center location, l^g, w^g, h^g are length ,width,height of the box, and θ^g is the yaw rotation around Z-axis. To retrieve找回 the ground truth box from a matching positive anchorparameterized as $(x_c^a, y_c^a, z_c^a, l^a, w^a, h^a, \theta^a)$,we define the residual vector残差向量 $\mathbf{u}^* \in \mathbb{R}^7$ containing the 7 regression targets corresponding to center location △x,△y,△z,three dimensions △l, △w, △h, and the rotation △Θ, which are computed as :

$$\Delta x = \frac{x_c^g - x_c^a}{d^a}, \Delta y = \frac{y_c^g - y_c^a}{d^a}, \Delta z = \frac{z_c^g - z_c^a}{h^a},$$
$$\Delta l = \log(\frac{l^g}{l^a}), \Delta w = \log(\frac{w^g}{w^a}), \Delta h = \log(\frac{h^g}{h^a}), \quad (1)$$
$$\Delta\theta = \theta^g - \theta^a$$

where $d^a = \sqrt{(l^a)^2 + (w^a)^2}$ is the diagonal对角线 of the base of the anchor box.Here, we aim to directly estimate the oriented定向的 3D box and normallize Δx and Δy homogeneously均匀的 with the diagonal对角线d^a,We define the loss function as follows:

$$L = \alpha \frac{1}{N_{\text{pos}}} \sum_i L_{\text{cls}}(p_i^{\text{pos}}, 1) + \beta \frac{1}{N_{\text{neg}}} \sum_j L_{\text{cls}}(p_j^{\text{neg}}, 0)$$
$$+ \frac{1}{N_{\text{pos}}} \sum_i L_{\text{reg}}(\mathbf{u}_i, \mathbf{u}_i^*) \quad (2)$$

where p_i^{pos} and p_j^{neg} represent the softmax output for positive anchor a_i^{pos} and negative anchor a_j^{neg} respectively,while $\mathbf{u}_i \in \mathbb{R}^7$ and $\mathbf{u}^* \in \mathbb{R}^7$ are the regression回归 output and ground truth for positive anchor a_i^{pos}.The first two terms are the normalized classification loss for $\{a_i^{\text{pos}}\}_{i=1\ldots N_{\text{pos}}}$ and $\{a_j^{\text{neg}}\}_{j=1\ldots N_{\text{neg}}}$, where the L_{cls} stands for binary cross entropy loss二元交叉熵损失 and α，β are postive constants balancing the relative importance. The last term L_{reg} is the regression loss, where we use the SmoothL1 function.

Efficient Implementation

Gpus are optimized优化 for processing dense稠密 tensor structures. The problem with working directly with the point cloud is that the points are sparsely distributed across sapce and each voxel has a variable number of points.We devised设计的 a method that converts the point cloud into a dense tensor structure where stacked VFE operations can be processed in parallel across points and voxels.

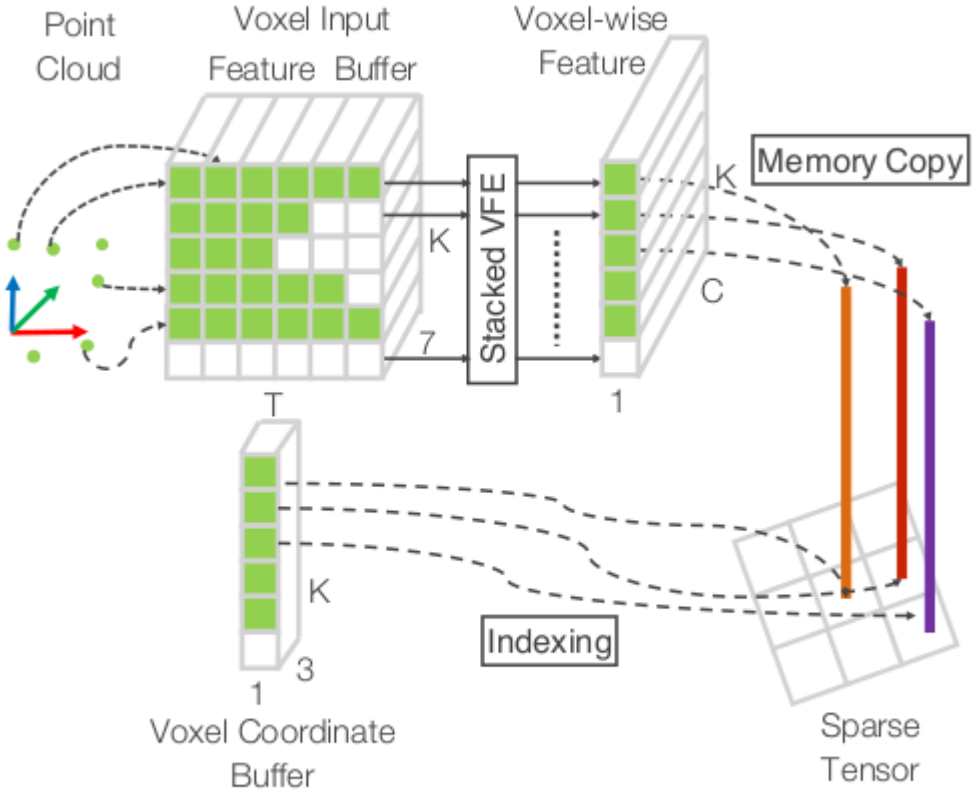


Figure 5. Illustration of efficient implementation.

The method is summarized in Figure 5. We initialize a KxTx7 dimensional tensor structure to store the voxel

input feature where K is the maximum number of non-empty voxels, T is the maximum number of points per voxel, and 7 is the input encoding dimension for each point.The points are randomized随机的 before processing.For each point in pointcloud, we check if the corresponding voxel already exists. This lookup operation is done efficiently in O(1) using a hash table where the voxel coordinate is used as the hash key.If the voxel is already initialized we insert the point to voxel location if there are less than T points, otherwise the point is ignored.If the voxel is not initialized, we initialize a new voxel, store its coordinate in the voxel coordinate buffer, and insert the point to this voxel location.The voxel input feature and coordinate buffers can be constructed建 via a signle pass over the point list, therefore its complexity is O(n).体素输入特征和坐标缓冲区可以通过对点列表的一次遍历来构造，因此其复杂度为O（n）。To further improve the memory/compute efficiency it is possible to only store a limited number of voxels(K) and ignore points coming from voxels with few points.

After the voxel input buffer is constructed,the stacked VFE only involves涉及 point level and voxel level dense operations which can be computed on a GPU in parallel.Note that, after concatenation级联 operations in VFE, we reset the features corresponding to empty points to zero such that they do not affect the computed voxel features.Finally, using the stored coordinate buffer we reorganize改组 the computed sparse voxel-wise structures to the dense voxel grid.最后，使用存储的坐标缓冲区，我们将计算的稀疏体素结构重组为密集体素网格.The following convolutional middle layers and RPN operations work on a dense voxel grid which can be efficiently implemented on a GPU.

Training Details

Network Details

Our experimental setup实验设置 is based on the LiDAR specifications of the KITTI dataset.
Car Detection For this task,we consider point clouds within the range of [-3,1]x[-40,40]x[0,70.4]meters along Z,Y,X axis respectively.Points taht are Points that are projected outside of image boundaries are removed.We choose a voxel size of $v_D = 0.4, v_H = 0.2, v_W = 0.2$ meters,which leads to D'=10,H'=400,W'=352. We set T = 35 as the maximum number of randomly sampled points in each non-empty voxel.We use two VFE layers VFE-1(7,32) and VFE-2(32,128).The final FCN maps VFE-2 output to $\mathbb{R}^{128}$.Thus our feature learning net generate a sparse tensor of shape 128x10x400x352.To aggregate voxel-wise features, we employ three convolution middle layers sequentially as Conv3D(128,64,3,(2,1,1),(1,1,1)),Conv3D(64,64,3,(1,1,1),(0,1,1)),and Conv3D(64,64,3,(2,1,1),(1,1,1)), which yields a 4D tensor of size 64x2x400x352.After reshaping, the input to RPN is a feature map of size 128x400x352,where the dimensions correspond to channel,height, and width of the 3D tensor.Figure 4 illustrates the detailed network architecture for this task.Unlike, we use only one anchor size, $l^a = 3.9, w^a = 1.6, h^a = 1.56$ meters, centered at集中于 $z_c^a = -1.0$ meters with two rotations, 0 and 90 degrees.Our anchor matching criteria匹配标准 as follows: An anchor is considered as positive if it has the highest Intersection over Union(IoU) with a ground truth or its IoU with ground truth is above 0.6(in bird's eye view).An anchor is considered as negative if the IoU between it and all ground true boxes is less than 0.45. We treat anchors as don't care if they have 0.45≤IoU≤0.6 with any ground truth.We set α＝1.5 and β=1 in Eqn.2.

Pedestrain and Cyclist Detection The input range is [-3,1]x[-20,20]x[0,48] meters along Z,Y,X axis respectively.We use the same voxel size as for car detection,which yields D=10, H=200, W=240.We set T=45

in order to obtain more LiDAR points for better capturing shape information.The feature learning network and convolutional middle layers ate identical相同 to the networks used in car detection task.For the RPN, we make one modification to block 1 in Figure 4 by changing the stride size in the first 2D convolution from 2 to 1. This allows finer resolution in anchor matching, which is necessary for detecting pedestrains and cyclists.We use anchor size $l^a = 0.8, w^a = 0.6, h^a = 1.73$ meters centered at $z_c^a = -0.6$ with 0 and 90 degrees rotation for pedestrain detection and use anchor size $l^a = 1.76, w^a = 0.6, h^a = 1.73$ meters centered at $z_c^a = -0.6$ with 0 and 90 degrees rotation for cyclist detecction.The specific anchor matching criteria is as follows:We assign an anchor as positive if it has the highest IoU with a ground truth, or its IoU with ground truth is above 0.5. An anchor is considered as negative if its IoU with every ground truth is less than 0.35.For anchors having 0.35≤IoU≤0.5 with any ground truth,we treat them as don't care.

During training, we use stochastic随机 gradient descent(SGD) with learning rate 0.01 for the first 150 epochs and decrease the learning rate to 0.001 for the last 10 epochs.We use a batchsize of 16 point clouds.

Data Augmentation 数据扩展

With less than 4000 training point clouds, training our network from scratch will inevitably 不可避免 suffer from overfitting. To reduce this issue, we introduce three different forms of data augmentation. The augmented training data are generated on-the-fly即时　without the need to be stored on disk.

Define set $\mathbf{M} = \{\mathbf{p}_i = [x_i, y_i, z_i, r_i]^T \in \mathbb{R}^4\}_{i=1,\ldots,N}$ as the whole point cloud, consisting of N points.We parameterize a 3D bounding box $\mathbf{b}_i$ as $(x_c, y_c, z_c, l, w, h, \theta)$,where x_c, y_c, z_c are center locations, l,w,h are length, width, height, and θ is the yaw rotation around Z-axis.We define $\Omega_i = \{\mathbf{p} | x \in [x_c - l/2, x_c + l/2], y \in [y_c - w/2, y_c + w/2], z \in [z_c - h/2, z_c + h/2], \mathbf{p} \in \mathbf{M}\}$ as the set containing all LiDAR points within $\mathbf{b}_i$, where p=[x,y,z,r] denotes a particular LiDAR point in the whole set M.

The first form of data augmentation applies perturbation摄动 independently to each ground truth 3D bounding box together with those LiDAR points within the box.Specifically,around Z-axis we rotate $\mathbf{b}_i$ and the associated Ω_i with respect to (x_c, y_c, z_c) by a uniformally统一的 distributed random variable $\Delta\theta \in [-\pi/10, +\pi/10]$. Then we add a translation (Δx,Δy,Δz) to the XYZ components of $\mathbf{b}_i$ and to each point in Ω_i,where Δx,Δy,Δz are drawn independently from a Gaussian distribution with mean zero and standard deviation 1.0. To avoid physically impossible outcomes, we perform a collision碰撞 test between any two boxes after the perturbation and revert还原 to orignal if a collision is detected.Since the perturbation is applied to each ground truth box and the associated LiDAR points independently, the network is able to learn from substantially实质上 more variations than from the orignal training data.

Secondly,we apply global scaling to all ground truth boxes $\mathbf{b}_i$ and to whole point cloud M. Specifically, we multiply the XYZ coordinates and the three dimensions of each $\mathbf{b}_i$, and the XYZ coordinates of all points in M with a random variable drawn from uniform distribution [0.95,1.05]. Introducing global scale augementation improves robustness of the network for detecting objects with various sizes and distances as shown in image-based classification and detection tasks.

Finally, we apply global rotation to all ground truth boxes $\mathbf{b}_i$ and to the whole point cloud M. The rotation is applied along Z-axis and around(0,0,0). The global rotation offset is determined by sampling from uniform distribution [-π/4,π/4].By rotating the entire point cloud, we simulate the vehicle making a turn.